KDF9 TIME-SHARING DIRECTOR

DIRECTOR DOCUMENTATION

This report contains a detailed description of the inner workings of the Time-Sharing Director. The report is provided as 'support documentation' in the sense described in section 2 of the KDF9 Service Routine Library Manual.

The report should be read in conjunction with the coding (which can be obtained from the current BINDIRECTOR - see Service Routine Library Manual) and the corresponding report on the Non-Time-Sharing Director. References are made to section A of the latter - by the letter 'N' followed by the appropriate section number.

The Director described in the Standard Time-Sharing Director and the report takes no account of optional extra facilities.

The Company reserves the right to change the information contained in this report without notice.

LIBRARY SERVICES,
SYSTEMS PROGRAMMING DEPARTMENT,
1ST MAY, 1965.

CONTENTS

1. Hardware Features

1.1 General

Before describing the special hardware features of the Time-Sharing KDF9 in detail a brief account of their objectives is necessary. The standard (non-Time-Sharing) KDF9 is already provided with interruption facilities which enable the operating program to be interrupted automatically, and then to pass control to the Director, when the program is held up by the action of a peripheral transfer either because it wants to use the peripheral unit involved or because it requires access to the area of core store which is locked-out for the duration of the transfer. However, there is no hardware which will record automatically which peripheral unit or which part of the store, is causing the hold-up. Nor is there any indication given at the end of a particular transfer that it was the cause of a hold-up.

The object of "Time-sharing" in KDF9 is to be able to have more than one program (besides the Director) stored in the machine at a time, and to run them on a "priority" basis, so that the highest priority program runs until it is held up (or ends) when control will be passed, via Director, to the next priority, which in turn runs until it is itself held up or ends, or until the conditions which caused any higher priority program to be held up, are removed, and so on. Thus, whenever a program is interrupted, one expects the Director to return control to one of the other programs, rather than waiting until the interrupted program can resume.

The existence of the KDF9 "Base Address" (BA) and the store and peripheral protection facilities provided by NOL (N2.1.3) and CPDAR (N2.1.4) make the storage (and movement) of several programs a safe and comparatively simple process. However, to implement efficiently the priority system described above, four extra hardware facilities are required, which make the difference between a "Time-sharing" and a "non-Time-sharing" KDF9:-

(a) a method of associating a priority with each program;

(b) a "Program hold-up" store to record in a dynamic manner which programs are held up at any time, and why;

(c) an RFI which will cause a program to be interrupted on the removal of a holdup condition which was causing any program of higher priority to be held up;

(d) a method of speedily interchanging the contents of the Nest, SJNS and Q-stores, for any two programs, so that the transfer of control from one program to another is as fast as possible.

The hardware implementation of these facilities, and the instructions which use them, are described below.

1.2 Priorities

Whenever any program is running it has associated with it a "current priority level" (CPL) in the range 0-3. This number is kept in a 2-bit register which is set by the instruction =K1 (N 2.1.3, N 2.1.1) which besides transferring D34-47 of N1 to BA and D24-33 to NOL, sets CPL equal to the value given in D34, 35 of N1.

Priority 0 is the highest priority, priority 3 the lowest.

It would be possible for several co-existent programs to have the same priority but this contradicts the principles underlying the design of the PHU store (see 1.3 below) and the multiple Nesting stores, so in practice there are up to 4 programs, each with its own priority number.

It is customary to refer to "Priority n" as meaning (in the right context): "the program which runs at priority level n". It is possible, subject to certain practical restrictions to alter the priority of a program.

The Director itself runs with priority 0 but this has no great significance. For reasons explained below (1.3), when the Director is about to execute a peripheral transfer on behalf of a program (i.e. involving that programs store area) CPL must be set to the value of that programs priority.

The actual significance of CPL is two-fold: firstly, any peripheral transfer has associated with it for its entire duration the value of CPL when it was called (this affects what happens in the hold-up case); and secondly, it determines whether or not the PR RFI (1.4) occurs when a peripheral hold-up is removed.

1.3 The Program Hold-up (PHU) store

This is a set of four 12-bit (D0-11) registers called PHU0, PHU1, PHU2 and PHU3, which are used to record, dynamically, peripheral holdups incurred by priorities 0,1,2 and 3 respectively.

When priority n is interrupted because of a peripheral hold up (LOV), PHU n automatically records the hold-up as follows: D11 is set to 1, and D10 is set to 1 or 0 according to whether the hold-up is due to a reference to a busy peripheral buffer, or to a locked-out core area. In the former case the busy buffer number is stored in D6-9 (D0-5 are cleared), in the latter case the least significant 10 bits of the 15-bit absolute address of the locked-out group are stored in D0-9. This is known as the "group address".

The PHU store is not altered if LOV occurs in Director mode, i.e. if the Director refers to a busy buffer or locked-out area.

PHU n is also set if priority n executes the instruction "INTQq", (N2.1.1) referring to a busy buffer. Again D11 and D10 are set to 1, the buffer number is stored in D6-9, and D1-5 are cleared: but D0 is set to 1.

Note that D11=1 in PHUn implies that priority n is held up by a peripheral transfer.

The end of a peripheral transfer which was called by priority n (i.e. called at any time, in Director or program mode, when CPL had the value n) will cause PHU n to be cleared if and only if:

(a) D0, 10 and 11 of PHU n are all non-zero
or (b) D10 and 11 are non-zero and the buffer whose number is given by D6-9 is not busy
or (c) D11 is non-zero, D10 is zero., and D0-9 give the absolute group address of a block of 32 words which is not locked-out.

This represents the removal of hold-ups due to (a) "Interrupt if Busy" - the holdup being removed at the end of any transfer called by priority n; (b) reference to a busy buffer; c) reference to a locked-out part of the store.

In simple terms, PHU n is set automatically when priority n is held up, and will be cleared automatically at the end of the appropriate peripheral transfer, provided that that transfer was called by, or on behalf of the same priority, i.e. when CPL was equal to n.

It is possible to ensure that PHU n will always be cleared automatically in the "lock-out" holdup case (c) by arranging, firstly, that no program ever transfers into another programs store area (with the proviso mentioned above, that when the Director starts a transfer involving a priority's area, that priority number must be in CPL); and secondly, that no program ever has its priority number changed while any part of its area is locked-out. Thus no priority can be held up by a lock-out set by another.

However it is impossible to ensure that no program ever refers to a buffer which has been made busy by another program, An obvious example is the "shared" magnetic tape buffer, which is made busy when any one of the units connected to it - which might be allocated to different priorities - is carrying out a transfer. Therefore, in order to avoid a situation which could involve a program being apparently held up for ever, the hardware is so arranged that at the end of any peripheral transfer called by priority n, not only is PHUn examined and, if need be, cleared, but also the remaining 3 PHU registers are examined, and if any of them are showing "busy" hold-ups due to a buffer which is no longer busy, the EDT RFI is set (EDT will of course be set anyway if the transfer was called in Director mode).

Note that these other PHU registers are not cleared, in this case. To do so would be dangerous: it is preferable that the Director should be made more positively aware of the situation, in order that it may decide whether or not to over-rule the normal priority rules in this case; to allow one of the lower priority programs to use the shared buffer, for instance. In a case where two programs are sharing a buffer, and the higher priority of the two makes frequent use of it while the lower priority only uses it occasionally, a judicial reversal of priorities, to allow the lower priority access to the buffer in preference to the higher, can lead to more efficient operation.

In order to have the necessary control over the PHU store, the Director must be able to examine its contents, and to clear the individual parts of it.

The instruction K5 causes the least significant 6 bits of each part of the PHU store to be brought into N1, nesting down one place. Thus,

```
        D6-11 of PHU0 occupy D0-5  of N1
        "  "  "  PHU1    "    D6-11  " "
        "  "  "  PHU2    "   D12-17  " "
        "  "  "  PHU3    "   D18-23  " "
```

Thus the Director can tell which priorities are held up at any time, which are held up by busy buffers, and in such cases, by which buffers.

The instruction CLOQq, as well as clearing lock-outs from the specified area, also causes PHUn to be cleared, where n is the value of CPL when the instruction is obeyed.

Note that the Director can determine when a program is held up by a lock out, but not the address involved (because it only sees 4 bits of this address).

The effect of all this is that Director must always look at the PHU store to determine which priorities can be resumed, i.e. are not held up: and also, whenever EDT occurs, Director must see whether any parts of the PHU have to be cleared, and, if so, whether the normal priority system should be over-ruled.

1.4 Program Ready RFI

If at the end of a peripheral transfer called by priority n, the procedures described above lead to the automatic clearing of PHU n: and if n is less than the value of CPL, an RFI called "Program Ready" (PR) is set.

In practice this means that if the hardware detects the end of a peripheral hold-up on a program which is of higher priority than the one currently operating, an interruption occurs.

This RFI corresponds to D22 in the copy of RFIR which the instruction K4 (N2.1.5) transfers to N1. When this RFI is detected the Director is expected to look at Dll of each part of the PHU store to see which priority can now resume.

The same process will follow the occurrence of the LOV interrupt. LOV corresponds to D28 in N1, after K4. In this case a change to a lower priority level can be expected.

The normal setting of PR is inhibited when the transfer whose ending led to the clearing of the PHU register is also found to have been the reason for "buffer busy" hold-ups at other priorities: in such cases EDT is set anyway, as described in 1.3 above.

PR is also set by the instruction "Interrupt if Busy" (INTQq) when the relevant buffer is found to be busy, in order to cause the required interruption. This avoids the need to have a separate RFI for this case. The same action is required of Director as in the normal PR or LOV interruptions.

1.5 Switching of Nests, etc.

To provide the necessary rapid changeover of the contents of Q-stores, Nests and SJNS's, four Sets of each are provided on the Time-sharing KDF9, together with instructions for switching from one to another.

The Sets are numbered 0-3 but this numbering is completely independent of the priority numbering system. Each Set comprises 15 Q-stores (1-15), a Nesting-store stack of sixteen 48-bit cells, and an SJNS stack of sixteen 16-bit cells. The two stack counters are not quadruplicated, nor are the overflow and Test registers.

In order to switch from one Set to another, it is necessary to obey the instruction =K3 with the following parameter in N1:

```
       D0,1  = New Set number
       D2-6  = Value to be put in the Nest Stack Counter
       D7-11 = Value to be put in the SJNS Stack Counter
```

The effect of the instruction is to bring into action the new Set specified by D0,1, and to replace the current values of the Nest and SJNS stack counters by the specified values. No nesting is involved in this operation.

The instruction =K3 should be followed by six DUMMY instructions to allow the new Set time to settle down.

Since it is only the stacks that are involved, the top 3 and 1 cells, respectively, of the Nest and SJNS are not affected. There fore whenever control is passed from one program to another, it is necessary, before obeying =K3, to insert 3 and 1 "dummy" items, respectively, into these positions, thus causing the (up to) 16 items in each which must be preserved to be nested down completely into the stacks. The dummy item in N1 must be the parameter for =K3. Between performing the interchange and returning to the new program, these dummy items have to be removed. Heading the (up to) 16 items which are preserved in the SJNS stack is the return address to the program.

Although the instruction =K3 does not alter the contents of either the "old" or "new" Set of Q-stores and stacks. it does cause the two stack counters to be overwritten. In order to preserve their values for future use, the instruction K7 is provided, which fetches into N1 the current "Set" number, and the current values of the stack counters, in the same format as for =K3. Here a slight complication is involved, because K7 nests down one place, but fetches the values of the stack counters before this nesting operation has been recorded. So to get a true record of the state of the counters after obeying K7, it is necessary to add D6 to N1 -- the equivalent of adding 1 to the Nest stack counter.

Illustrated below is a typical sequence for interchanging two Sets. Here, the initial and final states of the nests are exactly as they would be on entry to, and exit from, the Director.

(a) Insert one dummy link in SJNS.
(b) Insert two dummy items in Nest.
(c) K7; (nests down one place).
(d) Add D6 to N1, without nesting down further (by e.g. SHC+7; NOT; NEG; SHC-7;)
(e) Store N1 (old Set parameter) in a suitable location
(f) Fetch new Set parameter to N1.
(g) =K3; DUMMY; DUMMY; DUMMY; DUMMY; DUMMY;
(h) Erase three items from Nest.
(i) Erase dummy link from SJNS.

When adding corrections to the "stack counters" in N1 (as fetched by K7) it is important to ensure that no spill occurs from one "counter" to the other, or into the Set number position. This could only happen if NOUV had occurred.

The "Set parameter" used in the example is the record of the Set number and the stack counters actually used by =K3. It therefore shows excesses of 3 and 1, respectively, in the Nest and SJNS stack counters over the values these took just after the program they pertain to was interrupted.

Operations (e) and (f), above, would in practice probably refer to locations which depend on priority numbers, so the addresses used are likely to be modified by the contents of Q-stores. Care must be taken to ensure that those Q-stores have the correct values (their contents before interruption) restored to them before =K3 is obeyed.

1.6 Summary of Director-only instructions

This includes all the instructions of this type used by the Time-sharing Director, Some of these have greater power than they do in a non-Timesharing KDF9.

1.6.1 EXITD (N 1.2)

1.6.2 CLOQq (N2.1.1) As well as clearing the lockout specified by Qq, this instruction also clears PHUn, where n is the current value of CPL

1.6.3 CTQq (N2.1.2) Clear transfer

1.6.4 =K0. Switch buzzer on or off. There is a "buzzer" attached to KDF9 which can be switched on or off by obeying =K0 with the contents of N1 non-zero or zero respectively. No nesting. Thus, to switch off the buzzer, one obeys: ZERO; =K0; ERASE;

1.6.5 =K1. (N2.1.3) This instruction transfers D24-33 of N1 to NOL, D34-35 to CPL, and D38-47 to BA. No. nesting.

1.6.6 =K2. (N2.1.4) Set CPDAR. No nesting

1.6.7 =K3. Brings in a new Set of Q-stores, nests, etc., using a parameter in N1. No nesting
(D0, 1 = New Set number
D2-6 = Nest stack counter
D7-11 = SJNS stack counter)

1.6.8 K4. (N2.1.5) Fetch RFIR and Clock. Two significant RFIs are PR (D22) and LOV (D28). Nests down 1 place.

1.6.9 K5. Fetches l.s. 6 bits of PHU0, PHU1, PHU2, PHU3, to D0-5, D6-11, D12-17, D18-23, resectively, of Nl. Nests down 1 place.

1.6.10 K7. Fetches current Set number, and Nest and SJNS stack counter values, to N1, in format as for =K3. Note that the value of the Nest stack counter shown in D2-6 is 1 less than the value it actually has after obeying this instruction, because of nesting down 1 place.

2. Storage and input of programs

2.1 A and B levels

The Time-sharing Director can store up to four programs in the machine at once, each at a different priority level.

The four priority levels are divided, while the machine is running, into two types, A levels and B levels, and the programs in these levels are referred to a A programs and B programs. The levels to be used as A levels are specified when Director is input. Any number of A levels, from none to four, can be specified.

The A programs work in a °sausage machine° - when one finishes, any A programs lower in priority are upgraded automatically, and a new A program is called for into the lowest A level. B programs are only read in by a typewriter interrupt.

In order that, when any A program finishes, it should be possible to replace it immediately by any other -usually the next in a "queue" prepared by off-line or on-line scheduling - it is necessary that all A programs should operate within the same machine configuration, in respect of core storage and "unlabelled" peripheral units (including paper tape readers, punches, printers, etc., but excluding magnetic tape units, drum and OUT 8 output streams). The A program "Configuration" is supplied by the operators when the Director is loaded, as well as the priorities to be used by A programs.

The decision as to whether a given program is A or B is usually made off-line on the following basis. Suppose n A levels are specified when director is read in (n . 1,2,3 or 4. If n 0, the problem does not arise). Then a program is an A program, and so can be read into an A level, if there are enough core-store and peripheral devices (other, than magnetic tape units) for n such programs to run simultaneously on the machine. Thus if a machine has 16K of core-store, 2 p.t. readers, 2p.t. punches, 1 line printer and 1 card. reader, and if 2 A levels are specified, then any program needing not more than 1 pot. reader, 1 p.t. punch and 8K of core-store, and not needing a line printer or card reader can be an A program. Any other program will be a B program. It is entirely up to the Operator which category a program belongs to.

The devices that an A program needs are given ("pre allocated") to it when it is read in (the program still has to do OUT 5 to get them allocated to it), and remain with it the whole time it is in the machine. As soon as it is off the machine, these devices will be among those available for the next A program (provided they are not required by a higher priority), which is read in on the °sausage machine° principle described above Thus the whole time an A program is in the machine, the operators know precisely which devices it will be using.

2.2 Storage Assignment

The part of the core store which a program occupies - always a continuous area, and a multiple of 32 words beginning at an absolute address which is also a multiple of 32 - is said to be "assigned" to the program. The total area available for assignment to programs extends from the very top (highest address) of the store, down to the lowest word in the store which is not part of the Director, and whose address is a multiple of 32.

"A" programs are stored at the top of this area, and B programs at the bottom, next to Director. The Director constantly inspects the store assignments to ensure that programs are packed tightly next to one another, and to the limits of the available area. As a result of this it is necessary, from time to time, for the Director to physically move programs in the store. This always involves moving a program - that is, the whole of the area assigned to a program - into a vacant, unassigned, area next to it; there is no question of ever interchanging the locations of two programs, nor do programs "leapfrog" over one another. The effect is of "sliding" programs up or down the store, according to whether they are A or B, respectively.

This ensures that whatever space becomes vacant. as programs expire, accumulates in the middle, between the B and A programs. New programs are input to this area, known as the "Hole in the middle".

A "B" program, whose input at a nominated priority is initiated by a typewriter interrupt, will be read into the bottom end of the Hole. As soon as there are available a paper tape reader not required. by a higher priority, and 32 words of store in the Hole, this area is assigned to the program, and the A and B-blocks are read in subsequently the amount of store indicated by the store requirement in the B-block will be obtained by assigning any vacant storage as soon as it becomes available in the Hole (provided it is not required by A program input at a higher priority) until the requisite amount: has been assigned: then the C-blocks can be read in and the program will start.

During the delay that there may be whilst the program is waiting to obtain enough store, or between the input of C-blocks, the assigned area may be moved down the store to take over space vacated by other B programs which have terminated, in the way already described.

The priority nominated for the input of a B program may be one of those nominated for A programs. In this cases that priority ceases to be "A" for the duration of the program.

Director will call for the input of an A program when the following circumstances obtain:-

(a) a vacant A priority is available which is lower than all those currently occupied by A programs
and (b) the A program peripheral requirement can be satisfied.
and (c) there is enough space in the "Hole in the middle" to meet the pre-ordained A program storage requirement.

This required amount of store, from the upper limit of the hole downwards, is assigned before starting to read in the A program since the procedure described above for B programs, which starts with only 32 words assigned. Once the B-block has been read, this area may be increased (from the top downwards) if the actual store requirement is less than the amount assigned: the vacant space thus created will be transferred to the "Hole in the Middle" by the normal program-moving procedure.

Note that core storage ceases to be "vacant" as soon as it has been assigned for program input, and is therefore liable to be "slid" up or down the store while the program so occupying it is still input. However, an assigned area can only be moved when there are no peripheral transfers pending on it.

The case described above, when the store area assigned to a B program "expands", between reading the B and C-blocks, from 32 words to the requisite number, is the only one which ever involves increasing the amount of store belonging to a program: it is the only case where there is a vacant area, i.e. the "Hole in the Middle", to expand into. The area assigned to a program can be decreased by three means: firstly, in the case already described, where an A program's store requirement is less than the amount pre-assigned for reading it in; secondly, when a program obeys OUT 5; and thirdly, when a program obeys OUT 2. In the last two cases, the new program/section and its predecessor have the same priority, and the contents of any part of the store common to both old and new program/sections are unaltered (provided, of course, that in the case of OUT 1, they have not been overwritten by the latter). But the new program/section must not demand more store space than the old: if it does, failure occurs. If the new program/section has a blank store requirement, it is given exactly the same assignment as its predecessor.

The new program/section has the same priority, and the same A/B characteristic, as the old one, and, for an A program, is pre-allocated the same peripheral units.

Only one A and one B program can be in the process of input at any time - this includes the input of a new program/section after OUT 1, which may therefore have to wait if program input is already going on at another priority.

2.3 Housekeeping by the Director

Whenever any program is loaded it is assigned one of the four sets of stacks and Q-stores. Although a program may change its priority while it is in the machine, it always keeps the same set of stacks etc., and so it is convenient to identify each program with the number of its "Set". To avoid confusion with priority numbers, the "Sets" are conventionally called P, Q, R, and S. The bottom two bits of the octal representation of these characters - 60, 61, 62, 63 - give the actual number of the "Set".

Inevitably, a great many of the "housekeeping" parameters used by the Director are in groups of 4 consecutive words (quartets), one word for each program. Within each quartet the words are arranged in priority number order, so that the first word pertains to priority 0, the second to

priority 1, and so on. This arrangement is chosen (in preference to Program letter order) because in practice the priority number occurs more naturally and hence is a more convenient parameter for indirect addressing. In every case in which the priority number is used for indirect addressing (or for any other purpose) it is stored in M5, and so the convention is adopted in this report of referring to e.g. "V29P104M5" as meaning "the 4 words stored in V29P104, V30P104, V31P104 and V32P104, which have similar significance in relation to priorities 0, 1, 2 and 3, respectively".

The housekeeping parameters which are particularly relevant to the storage of programs in the machine are the single word V0P104, and the quartet V9P0M5.

The most significant 24 bits of V0P104 are assigned in groups of 6 to the four priorities so that D0-5, D6-D11, D12-17 and D18-23 correspond respectively to priorities 0, 1, 2 and 3. The bits for priority 0 have the following significance:-

    D5 = 1 if priority 0 cannot be entered for any reason (apart from
           a peripheral hold up)
    D4 = 1 if priority 0 is occupied by a pre-assigned "A" level
    D3 = 1 if priority 0 is occupied by an "A" program
    D2 = 1 if priority 0 is occupied by a "B" program. Obviously
           D2 and D3 cannot both be non-zero; and if D3 = 1, D4
           must be non-zero.

The corresponding bits of the other groups of six have the same significance in relation to priorities 1, 2 and 3.

The "hold-up" bits, D5, 11, 17, 23, allow for every possible "software" reason for not entering a priority, and thus indicate, for instance, that there is no program occupying a certain priority level, or simply that the priority is held up pending the allocation of a peripheral unit. Note that after obeying the instruction V0P104; =K5; the hold-up bits in the PHU store are combined with those in V0P104, so that the sequence of instructions

    V0P104; K5; OR;

will leave a complete indication in those bits of K5, of which priorities can be entered and which cannot.

The quartet V9P0M5 contains the Base Address, Number of Locations and Priority Number for each priority. Just before any program is entered the contents of the appropriate word is transferred, by executing =K1 (1.4.1) to the BA/NOL/CPL registers.

The quartet V9P0M5 therefore indicates the areas of store occupied by each priority. This includes storage which may be in the process of being "expanded" or "contracted" during program input - it may, for instance, amount to only 32 words, during the input of the B-block of a B program. The BA portion of any word of this quartet is zero if, and only if, there is no vestige of a program at that particular priority level. It is only when BA = 0 that D2 and D3 of the corresponding 6-bit group in V0P104

are both zero; as soon as a priority is assigned any storage at all, it acquires the status of either an A or a B program, and keeps the status just as long as it has a foothold in the store: i.e. D38-47 (BA) of V9P0M5, and D2-3 of the appropriate group in V0P104 are set and cleared together.

2.4 Program movement and priority interchange

One of the properties of KDF 9 Time-sharing is that programs may have their priorities interchanged, and may be moved about in the store.

The priority of a program may be changed either as the result of a TINT V, or following the priority upgrading which may occur at the end of an A program. A program may be moved in the store as a result of the ending of a program leading to the expansion of the "Hole in the Middle".

As explained in 1.3 above, it is dangerous to change the priority of a program while any part of its area is locked out, because it may lead to a peripheral hold-up being "stuck" in the wrong PHU register. Similarly, a program may not be moved in the store while any part of its area is locked out - a lock-out implies that a transfer is going on in that area and so the contents of the area are liable to change. Therefore if a reason for moving a program, or changing its priority, occurs while any part of the program area is locked out, the details of the required change are recorded (in the appropriate member of the quartet V9P0M5), and at the same time the program is temporarily prevented from resuming, using the appropriate bit in V0P104.

The impending change is recorded in V9P0M5 as follows:

    D0-2 = quantity to be added to priority number
    D14-23 = new value of BA.

The program is only allowed to resume when D0-23 are all zero.

Not only is the program held up when a priority change or store move is impending, but also the Director must be prevented from starting any peripheral transfer which involves the program's area, as this would prolong the duration of the lock-out on the area. Therefore before any peripheral transfer executed by Director, a check is performed that the transfer area does not lie within a region belonging to a program which is going to be moved or have its priority altered. Subroutine P111 performs this check, requiring as data the (absolute) addresses of the transfer. This subroutine also performs another important function - if the transfer is to be allowed to proceed, P111 alters the current CPL to that pertaining to the program whose area is involved. This meets a requirement mentioned in 1.3; again, to reduce the probability of a peripheral holdup getting "stuck".

When the priorities of two programs are interchanged the relative positions of the relevant numbers of all the parameter quartets have to be changed accordingly. All the quartets which have to be re-ordered

in this way are stored in the V-stores of P104, from V1 onwards. There are two other quartets which require a slightly more complicated re-arrangement: one is the quartet V9P0M5, in which the values of the

The manual reads V0P0M5 but this has to be a typo

priority number (D34-5) and the priority displacement (D0-2) have to be modified as well; the other is the quartet V24P0M5, which comprises the four "TAB" words, each containing the characters CR-LF, Case Normal, then sufficient TABs to ensure that any message typed out for the relevant program starts in the right column, the priority number and a space: here the priority number (D40-1) has to be modified. In addition, the appropriate sets of 6 bits in V0P104 have to be interchanged. Any priority numbers recorded in special locations (e.g. the priority level at which A or B program input is taking place, and the priority number of the last program interrupted) must be altered it necessary. The priority - dependent subprogram hold-up table (4.1.5) and subprogram numbers in the various peripheral transfer queues (4.1.3) also have to be modified appropriately, All these functions are performed by the subroutine P105, which carries out all priority changes and store moves, and also continually inspects the store and the state of the A programs to see what new moves and priority changes need to be instigated.

When a program's area is moved in the store (also by P105), the appropriate member of the quartet V9P0M5 has to be modified (D14-23 and D38-47). Also any "absolute" addresses, in the subprogram tables and the various peripheral transfer queues (see 4.105 and 4.1.3), which are affected, have to be altered. This is all looked after by P105, which is entered after every EDT,

The parameters contained in the various quartets of P104 are defined in the User-code listing. One quartet, V25P104M5, is of particular interest here. The "software hold-up" bit corresponding to any priority in V0P104 is set if, and only if, the corresponding member of the quartet is non-zero. The various bits in each word of this quartet, when non-zero, each indicate a different reason for a holdup, as follows:-

    D43 = 1 implies a store move and/or priority change impends
    D44 "   "   a hold-up due to the "OUT 8" subprogram
    D45 "   "   the program is suspended by TINT S
    D46 "   "   the program is absent, or being input or terminated
    D47 "   "   the main subprogram is active

The significance of D44 and 47 is explained in the sections dealing, with subprograms (4.1.5, 4.1.6, 5.3), The termination hold-up, D46, is described in 2.5, below.

The subroutines P113 and P114 are used respectively, to set and clear bits in V25P104M5, and at the same time set or clear the hold-up bits in V0P104, as appropriate.

2.5 Program Input and Termination

While a program is being input or terminated at any priority level, entry to it is inhibited by the setting of D46 in V25P104M5, which in turn sets the appropriate holdup bit in V0P104. This particular

hold-up covers all sorts of conditions, ranging from the complete absence of the program (indicated by a zero BA in the relevant member of V9P0M5, and no B or A "program present" bit in V0P104), through the first stages of program input (when the program, still anonymous, has a small foothold in the store), to termination by TINT A, OUT 0, OUT 1, or OUT 2.

These different conditions are indicated by the quartet V29P104M5, the various bits of each member of which having the following significance:

    D0 = 1   during termination of a program, indicates that the
               termination of "OUT 8" - type procedures is awaited. The
               main termination subprogram is held up at one stage until
               the subsidiary (OUT 8) subprogram has completed sequences,
               which lead to this bit being cleared.

    D1=0,D2=1 this combination of bits occurring during program input
               indicates an input holdup - due e.g. to lack of a program
               input device, or sufficient core store - such that the
               input process can be terminated by TINT A. If TINT A
               is performed while V29P104M5 contains this pattern, it is
               converted to:

    D1=1,D2=1 which causes the input procedure to be appropriately
               cancelled (if TINT A is performed while V29P104M5 is
               non-zero, but contains some other combination in these
               bit positions, it is ignored).

               The execution of TINT E (A or B as appropriate) while
               an A or B program is awaiting a program tape, i.e. while
               the member of V29P104M5 corresponding to that program has
               D1 = 0, D2 = 1, will cause D2 to be reset to zero.

    D47 = 1   during the input or termination of a program, indicates
               that it is a B program,

    D46 = 1   OUT 1 in progress

    D45 = 1   TINT A or OUT 0 in progress, or normal program input

    D44 = 1   OUT 2 in progress

While a priority level is unoccupied, and is not involved in program input, the relevant member of V29P104M5 contains 5 (D45 and D47 = 1) regardless of whether the priority level is A or B.

Program input is performed by the subroutines P120 and. P52 - the latter handles B program input (TINT T) in its initial stages and soon joins P120. Program input is carried out using the main subprogram (see 4.1.5) of the priority level concerned, so that effectively it is an activity having the same priority as the program it is loading. Advantage is taken of the restriction that not more than one A program and one B program can be in process of input at one time, in that only 2 sets of the parameters relevant to these activities need be stored: these

are kept in V0-2 of P120 (A programs) and P52 (B programs), as follows:

    V0 = priority level at which input is taking place
         (negative if inactive)

    V1 = program tape number (A or B)

    V2 = input paper tape reader (must be cleared as soon as
         PTR is finished with)

The instruction sequences in P120 for A or B program input are largely identical. A useful subroutine is P124, which, given the priority number in M5, sets the address of either V0P120 or V0P52 in M6 according to whether an A or a B program is being input at that priority level - i.e. according to whether D47 of V29P104M5 is 0 or 1.

The subroutine P121 deals with all CRNP failures, including those following OUT 1.

Program termination is performed by subroutines P15 (OUT 0), P16 (OUT 1) and P17 (OUT 2). P17 and P33 (TINT A) both lead into P15. Program failures are handled by P2, which also leads to P15 if the failure results in program termination.

3. Interruption processing

3.1 Paths through the Director

As in the non-Time-sharing Director (N.4) there are two main paths through the Time-sharing Director from the interrupt entry at word 0 to resumption of a program. Fig. 1 shows them in outline.

The coding for both paths comprises P0 of the Director.

The "standard preliminaries"mentioned in Fig. 1 are the instructions:

```
Q0; SHL+63; =+Q0;
SET B140000; =K1; ERASE;
```

which are stored in words 0 and 1 (by the Director Call Program).

Because both paths require the use of at least one Q-store, Q5 is dumped before the paths diverge. Similarly the state of the overflow and TR are recorded. In this context "recorded" implies that they are stored in the member of a quartet - V1P104M5, in this case - which pertains to the program interrupted (the priority of this program is to be found in V7P0). "Dumped" implies parking temporarily in a single location.

The Short path is followed if one of the RFI's LOV or PR occurs and no other. Any other RFI (CLOCK, FLEX, LIV, NOUV, EDT, OUT, RESET, or the abnormal absence of any RFI's at all) sends the Director down the Long path.

The Link is not dumped (as happens in the non-Time-sharing Director, N4.1) but no subroutine entry is made until NOUV has been dealt with, to avoid the risk of corrupting the contents of SJNS (N2.2.3).

A Director/Program marker (V28P0) is used to record whether the machine is in Director mode or not. It is examined, and set to zero, immediately after the "standard preliminaries". An interruption occurring while this marker is zero leads to a loop stops, typing

    FAILS
    FAILS etc.

(This loop stop also occurs if any other hardware malfunction occurs to make the Director behave in a logically inconsistent way).

3.2 The Short Path

Following the standard initial steps described above, the subsequent actions of the Director as it proceeds along the Short path fall into three stages:-

Stage (a): Accounting. This consists of augmenting the run time record of the program interrupted, and checking that its time limit has not been exceeded (if it has, the Long Path is entered).

Stage (b): The hold-up bits in the PHU store and in V0P104 are examined to determine which is the highest priority program not held up for any reason. If there is no eligible program, the Long Path is entered.

Stage (c): Once the return priority has been determined, the Director carries out functions in respect of the new program which are more or less complementary to those carried out in the initial steps and stage (a) in respect of the interrupted program - it updates the run time record, sets up overflow and TR appropriately, resets CPDAR, and finally restores the previously dumped Q5 and brings into action the new Set of stacks and Q-stores (first having recorded their erstwhile state in the right member of the quartet V13P104M5) before resetting (non-zero) the Director/Program marker, and obeying the sequence of instructions

```
*=K1; ERASE; EXITD;
```

to reenter the chosen program.

Note that, just as in the non-Time-sharing Director, not more than 3 and 1 cells of the Nest and SJNS, respectively, are used along the Short Path, and the RFIR is only inspected once.

Stage (a) is common to all paths through the Director. The Long path usually rejoins the Short path before, but sometimes after, stage (b).

3.3 The Long Path

The outstanding difference between the Long paths of the Time-sharing and non-Time-sharing Directors is that, whereas in the latter all sequences of instructions are constrained to use only 3 cells and 2 cells, respectively, of Nest and SJNS, in the former the contents of both Nest and SJNS are completely dumped so that their full capacity may be used. The adaptability which results from this more than compensates for the additional storage required for dumping. Note that there is no conflict here between the provision of 4 Sets of Nests, etc, and the idea of dumping the Nest and SJNS in the core store - the object of the former is to make the frequently-used Short path as fast as possible, while the latter is aimed at simplifying the coding of the comparatively rarely used, but much more complicated, Long path. This dump only holds the Nest and SJNS of one program (the one last interrupted), not of all four.

Altogether four Q-stores - 4,5,6 and 7 - are dumped, and may be used by the Long path.

There are three "entries" to the Long Path (see fig. 1), and two "ways out". The first step on each entry is to dump Q4, Q6 and Q7, and the contents of the Nest and SJNS (which will be the ones corresponding to the interrupted program, whose priority number is in V7P0), and the last step before leaving (to join stage (b) or (ci of the Short path) must be to restore Q4,6 and 7 and refill the (empty) Nest and SJNS. Thus within the region bounded by the dotted line in fig. 1, 19 and 17 cells, respectively, of Nest and SJNS may be used, with Q4, Q5, Q6 and Q7 - and, of course, overflow and TR.

It may be necessary at various points in the Long path to manipulate the contents of a program's Nesting store or. SJNS: this may not necessarily be the program which has just been interrupted, so these contents may be either in the "Dump" or actually In one of the 3 "Sets" which are not currently in use. In order to carry out these manipulations a multiple-entry subroutine, P29, is used. This requires as a parameter the priority number, in M5, of the program whose Nest or SJNS are to be manipulated. The entries are :

    1P29   Fetch one item from program's Nest
    2P29   Store one item into program's Nest
    3P29   Erase one item from program's Nest
    4P29   Clear program's Nest altogether
    5P29   Fetch one item from program's SJNS
    6P29   Store one item into program's SJNS
    7P29   Clear program's SJNS altogether.

Here "Nest" and "SJNS" refer specifically to the contents of those belonging to the nominated program. Fetching and storing have the usual erasing and nesting effects on them. The item transferred will be brought to, or taken from, N1 of the Director's Nest. The validity of each operation is indicated by the EXIT from P29 (EXIT 2 - valid, EXIT 1 - invalid).

Certain sequences in the Long path (OUTS and program failures) do not use P29 initially to manipulate the Nest and SJNS, but operate directly on the "dump". This is perfectly legitimate because (a) the operations are performed on the Nest or SJNS of the interrupted program, and (b) they are performed before there has been any "hold-up" in the instruction sequence from the time of interruption. However once any of these Long path activities are suspended for any reason, an resumption they are not entitled to assume that the contents of the "dump" pertain to the same program as before, because during the suspension other programs could have been re-entered, and the contents of the dump will always belong to the last one entered. P29 always knows whether the Nest or SJNS it wants is in the dump or not by comparing M5 with V7P0, which contains the priority number of the program occupying the "dump".

The three entries to the Long Path are, in order of possible occurrence, (A) immediately after the initial steps (3.1), if any RFI2s other than PR or LOV are found, (B) after stage (a) of the short path, if the time limit has been exceeded, and (C) after stage (b) of the Short path, if there are no eligible priorities to re-enter. Entries (A) and (B) are strictly "Phasel", but entry (C) may occur repeatedly in "Phase 2".

Entry (A) uses the subroutine P1 to dump the Nest & SJNS and Q4,6,7 unless NOUV has occurred, in which case the dumping is slightly different. P1 parks N1 onwards in V0P1 onwards, S1 onwards in V16P1 onwards, and the numbers of items in the Nest and SJNS in V32P1 and V33P1, respectively. The NOUV dumping procedure is similar, but only N1 and N2 are dumped, in V0P1 and V1P1 (the remaining Nest items are erased); and if the capacity of the SJNS has been abused, S1 is dumped in V34P1 and the remaining SJNS items in V16P1 onwards (with a marker in V6P12). Following a NOUV, the "dump" cannot be restored to the Nest and SJNS, so must be typed out, and cleared, before any "hold-up" in the instruction sequence occurs.

Entries (B) and (C) both start with Pl. Entry (A) is followed by accounting procedures (similar to stage (a) of the Short path) which take account of the possibility of a clock interrupt. The next step, at which entry (B) also joins, is to deal with program failures (using P2) and OUT (P3). **After this, P4 is entered if FLEX has occurred, P5 if EDT has occurred. Next (entry (C) joins here), P60 is entered to have another look at the RFIR. P60 corresponds more or less to the subroutine /RWR/ in the non-Time-sharing Director: it has only one EXIT, but leaves a marker in N1 which is non-zero if FLEX, EDT or CLOCK RFIes have occurred. If N1 is non-zero on this occasion, the Long path returns to the doubly asterisked step above

As soon as P60 has left a zero in N1, the Nest and SJNS, being empty, are refilled from the V-stores of P1, and Q4,6 and 7 are replaced. The Short path is now rejoined, between stages (a) & (b) if V14P0 is negative, otherwise between stages (b) and (c). V14P0 is a marker whose contents, if non-negative, are a "return priority number" generated by P5 (after an EDT interrupt) for reasons connected with the forcible clearing of PHU registers (see 1.3)

D24 of the B.I. word holds a Parity indicator, and D25-47 the block count used by brick 1.

4.1.2 Action of P105

Some mention of the actions involved in store consolidation and priority interchange was made in 2.4. These actions are dealt with by P105, which performs the following steps:-

(1) Examines and clears a marker (V0P105).and skips step (2) if it was zero.

(2) For each priority in turn, effects any specified store movement (but only if the area involved is not locked out) and any specified priority interchange (but only if neither of the areas assigned to the specified priorities are locked-out).

"Store movement" involves not only moving the program in the store and altering its BA (in V9P0M5) but also modifying any absolute addresses referring to locations within the "moved area" which may be stored as parameters in any of the "Queues" (see 4.1.3) or any of the Subprogram hold-up parameter stores (see 4.1.5).

"Priority interchange" involves the following:-
- interchanging bits of V0P104.

- interchanging quartet members in the V-stores of P104, and (with additional complications) in V9P0M5 and V24P0M5.

- altering, if need be, the priority numbers recorded in V7P0, V0P52 and V0P120, and similar locations.

- interchanging the sets of subprogram parameters for the pairs of subprograms pertaining to the priorities involved (see 4.1.5).

- altering accordingly the subprogram numbers contained within those "Queue" transfer parameters which are holding up any of the affected subprograms (see 4.1.3, 4.1.5).

(3) Re-sets up all the priority interchanges required by the A-program concept, regarding only those A priorities which are neither occupied by B programs, nor already committed to inter change with B, or B-occupied, priorities, as eligible for interchange. Any new priority interchange set up in this step also causes V0P105 to be set non-zero.

(4) If there are gaps in the store, either between two A-programs or between the top of the store and the nearest A-program, sets up the store move which will close up the gap nearest the top of the store. Any new move set up causes V0P105 to be made non zero.

(5) Carries out a similar service for B-programs as (4) does for A-programs, relative to the bottom of the store.

(6) If V0P105 is now non-zero, returns to step (2).

(7) For each priority, causes the hold-up bit D43 of the correspond ing member of V25P104M5 to be set or cleared according to whether or not there is still an outstanding store move or priority interchange at that priority. If there are any out-standing, V0P105 is left non-zero.

4.1.3 Queued transfers

The format of transfers in any queue (e.g. the Flexowriter queue, dealt with by P59, the OUT 8 tape queue dealt with by P155, or the magnetic drum queue dealt with by P143) is usually one 48-bit parameter word per transfer, with a word consisting of all ones immediately after the last parameter word to indicate the end of the queue.

Because of the number of different types of transfer it has to cater for, the Flexowriter queue is more interesting than others. The three types are:-

(a) Queries: the absolute initial and final addresses of the query are stored in D16-31 and D32-47 of the parameter word. D0 is 1, D1 to 5 are all zeros, and D12-15 contain the number of the subprogram (0-9 - see 4.1.5) which called the query. The sub-

program number is required because a query, unlike the other types of Flexowriter transfer, holds up whatever process calls it, in order to wait for the answer, and therefore that process has to be part of a subprogram. P59 must remove this hold-up when the query has been answered and the answer checked, and to do this it must know the subprogram number.

(b) Parametric transfers: the parameter word has D0-15 = all ones, D16-31 initial (absolute) address, D32-47 final (absolute) address.

(c) Message words. A transfer of this type causes the <u>actual contents of the parameter word</u> to be typed. Any parameter word which does not fall into one of categories (a) or (b) is of this type. Clearly it is not permitted to insert into the Flexowriter queue any message word which starts with the character (octal) 40 - this would make it look like a Query parameter word - or with the characters (octal) 7777 - this looks like a parametric transfer or, if the word consists of "all ones", like the end of the queue (for this reason the "dummy" second word of one-word tape identifiers is always typed out as D0-2=0, D3-47=all ones, rather than the "all ones" form it takes in the non-Time-Sharing Director).

Another implication is that if any Flexowriter message is inserted into the queue in the form of a string of message words, all these words will be typed out, even if some word before the last one contains an EM; although each individual word is typed to EM. Hence the character octal 75 acts as "End Word" rather than "End Message" in this context.

A trap which it is easy to fall into is to try and use the same area of store for a message which one wishes to have typed out in slightly different forms (e.g. varying a priority number) on different occasions, as a query or parametric transfer. This may lead to the message being altered, and re-inserted into the queue, at a time when a similar parametric transfer, of the previous version of the message (now corrupted), is still waiting to be typed further up the queue. Such messages must either be typed as strings of message words, or must exist in their different form: in different locations (e.g. the "REACT" queries in P2).

Subroutines exist for inserting parameter words into the Flexowriter queue. They are:-

(a) P8 - insert query. On entry, N1 contains the final absolute address, N2 the initial absolute address and N3 a "number of extra links to be preserved" parameter identical to that required by entry 2 to P10 (see 4.1.6). This subroutine, like P10, requires the appropriate subprogram parameters in Q4 and M5, and like P10 it suspends the subprogram, which is not re entered until the query has been typed, answered, and checked.

(The subroutine P140, for inserting items into the Magnetic Drum queue, is similar).

(b) P7 - insert parametric transfer. On entry, N1 contains the final absolute address, N2 the initial absolute address.

(c) P6 - insert a string of parameter words, given in N2 (the first), N3, N4 etc. into the Flexowriter queue. The number of words is given in N1.

The Flexowriter queue is contained in the V-stores of P6. The only "tight loop" which can occur in Director is when the Flexowriter queue gets filled up, and the Director sticks in P6 until all the parameter words have been put in the queue.

P6 is used by P7 and P8, but otherwise usually only for inserting strings of message words.

As mentioned in 4.1.2, the parameter words in any queue may have to be modified by P105 because they contain absolute addresses referring to moved areas, and/or subprogram numbers altered by a priority interchange.

4.1.4 Action of P120

All that P120 does, when it is entered following EDT, is to examine all the priority levels to see if there is one which is suitable for A-program input. To be suitable, it must be an unoccupied A-level, and there must-be no lower priorities occupied by A-programs.

If a suitable priority cannot be found, P120 takes no further action. However, when it does find one, P120 initiates the main subprogram of that priority (see 4.1.5) before returning control to P5, so that it will be re-entered by P11.

At this stage no peripheral or store reservation has been made for the A-program. It is possible that the subprogram set up by P120 will not be re-entered: this could occur if TINT T is obeyed, calling for the input of a B-program at the same priority, before this subprogram can be re-entered - in such a case P52 (TINT T) would overwrite the subprogram parameters, left by P120, with others of its own.

Assuming the subprogram is re-entered, it checks first that there is no A-program input at any other level, by examining V0P120. Then it checks that the required complement of core storage and peripheral devices are available. Only after ensuring that these requirements can be met (an activity having the same priority as the program to be input) does the subprogram reserve the required amount of store, pre-allocate the required peripherals, set the "A-program present" bit in V0P104, and the priority number in V0P120, prior to continuing with the program input process. Otherwise the subprogram is closed down.

4.1.5 Subprograms - P11

The subprogram system of the Time-Sharing Director bears superficial resemblances to that of the non-Time-Sharing Director (N.6.2) but is more flexible and powerful, mainly because it can exploit the availability of the full depth of the SJNS. It was devised by A. Doust.

There are ten subprograms. Each represents an activity within the Director which can be temporarily suspended for one of the following reasons:-

(a) the subprogram requires access to a busy peripheral device or a locked-out area of store, or

(b) the subprogram is waiting for the end of a "queued" peripheral transfer, or

(c) the subprogram is waiting to be allocated a magnetic tape with a specified identifier, or

(d) the subprogram is waiting to be allocated some other type of peripheral unit.

The ten subprograms are numbered 0-9. Subprograms 0 and 1 belong to the Director - the former is used by P4 for typing out the "TINT" query, the latter is used by OUT 8 and by TINT M. The remaining eight are allocated to the four priorities: thus, subprograms 2p+2 and 2p+3 belong to priority p. The first of the two belonging to any priority, known as the "Main" subprogram, is used for activities which require the program at that priority level to be held up - e.g. allocation of peripheral units, program input - while the second subprogram, which is used for such activities as OUT 8 and magnetic drum transfers, does not automatically hold up its program. Usually it relies on ordinary core store lock-outs to hold up the program if needed, but it can also use D44 of V25P104M5 (see 2.4).

V1-40 of P11 contain a table in which the states of all the sub programs are recorded. The action of P11 is to examine this table, looking at the entries corresponding to all the subprograms in increasing numerical order (thus giving the correct relative priorities to subprograms 2-9). Any subprogram whose suspension can be terminated is immediately re-entered, and remains in control until for some reason it returns control to P11, which then resumes the examination of the table at the entry for the next subprogram. P11 ends when all the subprograms have been dealt with in this way.

Each subprogram owns 4 words in the table, known as SHUL, SHUP, SHUP* and SHUL*. For subprogram n, these are respectively V(n+1), V(n+11), V(n+21) and V(n+31) of P11. While a subprogram is suspended, D32-47 of its SHUL contains a link which, when placed in SJNS and EXIT 1 is obeyed, will cause the subprogram to be re entered at the correct point. SHUL* contains up to 3 extra items from the SJNS which can be preserved while the subprogram is held up, and which will be restored by P11 to the SJNS before the sub-

program is re-entered. SHUP and SHUP*, and D0 of SHUL, contain parameters which indicate the reason for the suspensions, as follows:-

Type of Suspension	D0 of SHUL	SHUP	SHUP*
(a) Peripheral	0	D0-15=Busy unit, or zero D16-31, 32-47 locked-out area (absolute addresses)	Unused
(b) Queued transfer	0	D0=1	Unused
(c) Identifier required	1	First word of identifier	Second word (D3-47 if "one word")
(d) Unit required	1	D0=1, D1-41=0, D42-47 = type number	Unused

SHUP* may be used by any subprogram to hold private parameters except when an "identifier required" suspension occurs.

The items in the table may have to be modified and interchanged by P105 (see 4.1.2).

Q4 is used to address this table. While dealing with subprogram n, its contents are:-

C4 = n
I4 = 1
M4 = AV(1+n)P11

When subprogram n is re-entered from P11, Q4 will contain these parameters, and the subprogram must ensure that when it subsequently returns control to P11, Q4 is in status quo, so that P11 may resume its examination of the table at the right point (though this rule, as will be seen below, has its exceptions).

When examining the entry in the table for any subprogram, P11 acts as described below:-

- if SHUL is zero: the subprogram is ignored. SHUL=0 indicates it is <u>closed down</u> and inactive.

- if SHUL is positive: the subprogram is entered if, and only if, SHUP is positive, the store area defined by D16-47 of SHUP is not locked-out, and the unit whose number (if non-zero) is in D0-15 of SHUP is not Busy. The contents of SHUP are left in Q7 when the subprogram is re-entered.

  Since words 0-31 of the Director are never locked out, re-entry to the subprogram is guaranteed if SHUP is zero. This fact is made use of when setting up subprograms, and also at the end of any

  "queued" transfer which is holding up a subprogram: in this case the appropriate subroutine (e.g. P59, P143) will clear the SHUP associated with the subprogram whose number is stored with the transfer parameters (see 4.1.3).

- if SHUL is negative: the subprogram is entered if the first octal character of SHUP is 40 (N.B. no tape identifier can start with this character),and if there exists a "U" unit of the type whose number is given in D42-47 of SHUP. The unit number is left in C6 and M6, and its descriptor is unaltered. If the first octal character of SHUP is not 40, SHUP and SHUP* are assumed to contain a required tape identifier (if this is one-word, SHUP* contains (octal) 07 77 77 77 77 77 77 77.) In this case, the subprogram is entered if, and only if, V0P101 is non-zero - this is the marker indicating that a new tape identifier is present (see 4.1.1) - with the contents of SHUP and SHUP* left in N1 and N2 respectively. It is up to the subprogram to see if it is the one required, using P58 (see 4.1.6).

When it is about to re-enter subprogram 2n or 2n+1, P11 first sets the value (n-1) in M5 (for subprograms 2-9, this is the corresponding priority number). If the subprogram is 2, 4, 6 or 8, D47 of V25P104M5 is <u>cleared</u>. Finally a succession of items are transferred to SJNS:

- a link which when used by either EXIT 1 or EXIT 2 will return control to the correct point in P11 - this makes the subprogram look like a subroutine of P11.

- <u>if</u> SHUL* is non-zero, D32-47 of SHUL*

- <u>unless</u> D0-31 of SHUL* are all zero, D16-31 of SHUL*

- <u>if</u> non-zero, D0-15 of SHUL*

- D32-47 of SHUL

(at least 2, and not more than 5, items)

SHUL is now cleared, and the subprogram is entered by obeying EXIT 1.

By virtue of this clearing of SHUL (and also the clearing of D47 of V25P104M5 if it is the main subprogram of a priority - see 2.4), the subprogram will automatically have been closed down when it returns control to P11, unless it has reloaded SHUL. The loading of SHUL, SHUP, etc., is usually carried out by P10, and is described in the next section

4.1.6 Subprograms continued - P10 etc.

The actions of P11 in respect of a subprogram which it causes to be re entered are, besides entering it at the correct location (SHUL), to transfer a link, and the non-zero items in SHUL*, to the SJNS, and also to leave the contents of SHUP, and sometimes SHUP*, in suitable places. The subroutine P10 exists largely to perform the opposite functions, that is, when a sub program is to be suspended, to load SHUL, SHUL* and SHUP with the correct parameters (SHUP*, as will be seen, is dealt with by P58 only), and then to return control to the routine (usually P11) of which the subprogram is a "subroutine".

P10 has several entry points, of which one (the "normal" one, JSP10) is conditional, in the sense that the subprogram is only suspended if a peripheral holdup is present: all the other entries are guaranteed to suspend the subprogram. The following requirements apply to all entries:

(i) the correct parameters must be in Q4, and when the subprogram is number 2,4,6 or 8, the correct priority number must be in M5 (to address V25P104M5, etc.).

(ii) entry must be made using a "jump to subroutine" instruction. If the subprogram is actually suspended, the link thus planted will be stored in SHUL, so that when the subprogram is re-entered (and this applies even if it is not suspended) it will be at the instruction immediately after the "jump to subroutine".

(iii) a parameter must be supplied which states how many items (up to 3) are to be transferred from SJNS to SHUL* if this entry causes the subprogram to be suspended.

(iv) if the subprogram is suspended, P10 is left (after having deposited the appropriate items in SHUL, SHUL* and SHUP) by obeying EXIT 1 or EXIT 2, and a further parameter must be supplied to say which of these two alternatives if required. This parameter, and the number of items n (0≤n≤3) which are to be transferred from SJNS to SHUL*, are combined in. one word, referred to below as the "link parameter", which takes the value n or ñ (i.e. -1-n) respectively, according to whether EXIT 1 or EXIT 2 is to be obeyed when leaving P10.

Thus if P10 is entered with a link parameter of 2, and the subprogram is suspended, the overall effect on the SJNS after P10 has finished is that 3 items have been erased: the top two go to SHUL*, and the third is used when EXIT 1 is obeyed to leave P10. The link planted on entry to P10 does not count as it goes straight to SHUL. When the subprogram is re-entered subsequently by P11 the SJNS will contain (starting at the top and working down): the 2 items from SHUL*, a link to return control to P11, and links to return control from P11 to P5 and from PS to the long path.

Note that if a subprogram enters P10 but is not suspended, it will continue at the next instruction after the "jump to subroutine" with the SJNS, SHUL, etc., unaltered.

The use of the SJNS to determine where control shall be transferred when P10 is left, means that P10 can be used to initiate a subprogram, i.e to suspend an activity in such a way that it can be re-entered as a sub program by P11: provided that this initiation, or initial suspension, takes place within a subroutine, so that there is in SJNS a link which may be used, by EXIT 1 or EXIT 2, when transferring control from P10 after the latter has set up SHUL, SHUT,* and SHUP. Because there may be some significance in whether this link is used by EXIT 1 or EXIT 2, the choice must be specified in the link parameter.

Once the subprogram has been initiated, and re-entered by P11, any subsequent suspension by P10 will cause control to revert to P11, via the link planted by the latter - i.e. the subprogram behaves like a subroutine

of P11. Whether EXIT 1 or EXIT 2 is used to return control to P11 is immaterial, as either will reenter P11 at the point where it updates Q4 and goes on to examine the next subprogram.

Whenever P10 suspends subprogram 2,4,6 or 8, it also sets D47 of the appropriate member of V25P104M5 so that during its suspension, the priority of which this is the main subprogram is also held up.

P10 is not used to close down a subprogram, because, as explained in 4.1.5, when a subprogram is re-entered its SHUL is cleared. All that a subprogram needs to do to close itself down is to obey EXIT 1 or EXIT 2 using the link planted by P11.

The full power of the P11-P10 combination will be shown in the examples of its uses in connection with FLEX and OUT interrupts.

Entries to P10 are as follows:-

(a) JSP10: obeyed with the link parameter in N1.. The subprogram is suspended if, and only if, the store area defined by the absolute addresses in I7 and M7 is locked out, or (if C7 is nonzero) if the peripheral unit, whose number is in C7, is busy. The suspension will be the first type of subprogram hold-up, i.e. D0 of SHUL is made zero, and Q7 is transferred to SHUP, by P10. When the subprogram resumes, at the instruction after JSP10, the link parameter will have been erased from N1 and Q7 will be in status quo.

(b) JS2P10: obeyed with the link parameter in N3. N2 contains a quantity which P10 will OR with the link before transferring the latter to SHUL. N1 contains the quantity to be transferred by P10 to SHUP. Suspension is unconditional. P10 will erase N1, N2, N3. When the subprogram is reentered by P11 (at the instruction following JS2P10) the disposition of items in Q7, the Nest, etc., will of course depend on the type of suspension determined by D0 of SHUL, and SHUP (see 4.1.5).

(c) JS3P10: obeyed with the link parameter in N1. This entry is exactly equivalent to obeying ZERO; Q7; JS2P10; (i.e. like JSP10, with suspension guaranteed).

(d) JS4P10: exactly equivalent to obeying ZERO; ZERO; ZERO; JS2P10; This is a simple method of suspending, or initiating, a subprogram so that its re-entry next time P11 is entered is guaranteed, since D0 of SHUL=0 and SHUP.O. SHUL' is also cleared.

Apart from the link parameter, the entry JSP10 is the almost exact equivalent of the JS/RWL5/; used in the non-Time-Sharing Director (N6.2).

Entry 2 (JS2P10) is so flexible, permitting any type of suspension to be set in SHUL and SHUP, that it is used to set up almost all the subprogram hold-ups implicit in other subroutines, e.g. P8 and P58. P8, the subroutine to set a typewriter query in the flexowriter queue (4.1.3), requires the link parameter in N3 on entry, and its last instruction, after inserting the transfer parameters in the queue, is J2P10. P58 is a subroutine used by OUT 4 and OUT 10 which, given a 2-word identifier in N1

and N2, will suspend the subprogram in the correct manner if the required tape is not available. To do this it will obey:

```
REV;   (interchange 2 words of identifier)
=E20M4; (store 2nd word in SHUP*)
SET1; (link parameter - to preserve link for P58 in SHUL*)
ZERO; NOT; STR; NEV; (sets D0 in N1)
CAB; (1st word of identifier to N1, for SHUP)
JS2P10; (to hold up subprogram - SHUL has D0=1)
```

When the subprogram is reentered by P11, which will occur when the new identifier marker, V0P101, is non-zero, the identifier will be transferred by P11 from SHUP and SHUP* to N1 and N2. The link for P58 will be in SJNS, so that the position is exactly as it was when P58 was first entered. The next instruction is therefore: JP58;

Note that a subprogram may address its SHUL, SHUP, SHUP*, SHUL*, as E0M4, E10M4, E20M4 and E30M4, respectively.

4.2 FLEX. The action of P4

The subroutine P4 is entered from the long path if FLEX has occurred, i.e. if V16P0 is found to be non-zero. Its first action is to set V19P0 (the EDT marker) non-zero, to ensure that by pressing the Interrupt key on the Flexowriter the operator can force an "artificial" EDT.

Subprogram 0 will be used to type the query "TINT". If this subprogram is found to be active (i.e. its SHUL word, V1P11, is non-zero) P4 exits to the long path without clearing V16P0 - this will cause the Director to cycle round the inner loop of the long path until the FLEX RFI can be dealt with.

If V1P11 is zero, P4 sets up Q4 with the correct parameters for subprogram 0 (a copy of V0P11) and enters P8 to put the parameters for typing the "TINT" query into the Flexowriter queue, with a link parameter of zero. As explained in 4.1.6, this will cause SHUL (with D0=0) and SHUP (with D0=1) for subprogram 0 to be set up, and will then cause control to be returned to the long path when P8(P10) obeys the link (using EXIT 1) which was planted by the entry to P4. The subsequent actions of P4 therefore take place in subprogram 0, following re-entry by P11 after the query has been answered (giving EDT automatically) and checked.

The reply to the query is checked to ensure it is a letter in the right range. If it is not, the query is typed again (preceded by AGAIN): this time, P8 transfers control back to P11 rather than to the long path. When a satisfactory answer is received, a branch is made depending on the value of the letter typed. This branch is made to look like a subroutine jump, so that in effect each TINT is dealt with by a subroutine of P4. These subroutines are:

TINT A - P33	
" B - P34	TINT M - P45
" E - P37	" P - P48
" G - P39	" R - P50
" H - P40	" S - P51
" I - P41	" T - P52
" J - P42	" U - P53
" L - P44	" V - P54

and when each is entered in this context, there will be four links in SJNS: the links to return control to P4, from P4 to P11, from P11 to P5, and from P5 to the long path.

Each of these subroutines has two exits. EXIT 2 is the normal exit for a successful execution. EXIT 1 is obeyed when any check fails and causes the TINT query (preceded by AGAIN) to be repeated.

To illustrate the ways in which these TINT subroutines exploit P10 and the subprogram system, P33, P41, P45, and P52 are discussed below in some detail. The other TINT subroutines are quite straightforward, and examination of any of them will show that they work in a simple way, and finish by obeying either EXIT 1 (failure) or EXIT 2 (successful execution). Following the successful execution of any of these subroutines, Q4 is liable to have been altered, so the instructions to which EXIT 2 leads are:

V0P11; = Q4; EXIT 1;

which reset Q4 to the value required by P11, and then return control to P11, leaving subprogram 0 closed down. The value of M5 is immaterial when P11 is resumed.

The subroutines P31 (fetch character) and P32 (fetch octal number) are much used by P4 and by the TINT subroutines (they are analogous to /RFC/ and /RFN/ in the non-Time-Sharing Director - see N5.3). Also useful is P56, which extracts a program letter from C7 (where it will have been left by P31) and sets the corresponding priority number in M5.

4.2.1 P33 - TINT A (Terminate program)

P31 and P56 are entered to get the priority number of the program concerned into M5. V29P104M5 is examined to see whether the program is in process of termination (V29P104M5 nonzero) or not. If the program is already being terminated, P33 merely sets a 1 in D1 of V29P104M5 if D2 of that word is nonzero (see 2.5), and then ends EXIT 2.

If the program is not already in process of termination, TINT A must end it. To do this, it first clears D45 of V25P104M5 (see 2.4) and then, by obeying JS2P10 with a link parameter of -1 (which makes P10 clear SHUL*, and end by obeying EXIT 2) sets up the main subprogram of the priority to be terminated, closing down subprogram 0 as shown below.

The steps performed by P33 in doing this are:

JSP30; (this most useful subroutine sets up in Q4 the parameters appropriate to the main subprogram of the priority whose number is given in M5)
ZERO; NOT; (link parameter)
ZERO; (to ensure D0 of SHUL=0)
C7; (the character following the program letter; if it is +, the TINT A procedure is different. Here it is masquerading as the final absolute address (in the range 0-63) of a potentially locked-out area, and will thus be stored in SHUP. Since locations 0-63 are never locked-

out this is more or less legitimate).
JS2P10; (this causes SHUL and SHUP of the priority's main subprogram to be set up as indicated by the parameters in N1 and N2. SHUL* is cleared and P10 obeys EXIT 2 to return control to P4. P4 now restores Q4 to its original value, for subprogram 0, and obeys EXIT1 to return control to P119 Which will soon come to deal with the subprogram newly set up. Subprogram 0 is left closed down since its SHUL is still zero.

When the new subprogram is re-entered by P11, M7 will now contain the character following the program letter. If this is + (octal 35), an ending number(4) is placed in V0P29 and P2 entered at reference 2 (see 5.1), causing the Nest and SJNS to be typed out before entry to P15 which sets off the normal termination procedure. Otherwise P15 is entered direct.

Note that TINT A does not check whether the priority's main subprogram is suspended or closed down, before taking it over. This characteristic is shared by TINT I (see 4.2.2). It implies that any activity carried out by the main subprogram while its parent program is not being terminated must always suspend itself in such a way that it does not matter if it is closed down (because of program termination or restart) during any suspension.

4.2.2 P41 - TINT I (Restarts)

Like P33, P41 uses P31 and P56 to set up the appropriate priority number in M5, and then examines V29P104M5: action is only taken if the latter is zero. This action consists of adding an extra link to the program's SJNS, causing it to resume at syllable 0 (even restart) or 3 (odd restart) of word E4. In the former case, P41 clears the SHUL of the priority's main subprogram, and removes D47 of V25P104M59 by obeying:
JSP30; ZERO; =M0M4; SET 1; JSP114;

4.2.3 P45 - TINT M (Store Post-Mortem)

TINT M has to carry out two operations which may cause hold-ups: it must allocate itself (the Director) a paper tape punch, and it must output a sequence of messages on that punch. It uses subprogram number 1 for these activities, and accordingly sets up Q4. It may, or may not, be suspended, in the process of obtaining the paper tape punch, and so the first suspension could occur then, or during the output of paper tape: for this reason, a link parameter of -1 has to be used both for suspending the subprogram if a paper tape punch is not available (in which case SHUL is negative and SHUP = D0+D47: JS2P10 is used) and also for the conditional suspension (JSP10) occurring during the actual punching of paper tape.

The first suspension closes down subprogram O. Care must be exercised when using P45 because it does not check that subprogram 1 is already in use before using it itself.

4.2.4 P52 - TINT T (Input B program)

P52 must read in the priority number at which the B program is to be input, and check that it is not occupied or involved in a priority swop, and that B-program input is not proceeding at any other level. Other checks must be performed on the "operator's store limit", if present, and also on whether 32 words are available for reading the B block. If all the checks are passed, P118 is used to set up the Base Address in V0P104, and the priority number in V0P52 (indicating which priority is involved in B program input). Then the main subprogram of that priority is set up (just as P120 sets up this subprogram when initiating the input of an A program - see 4.1.4).

A point of interest about the way this subprogram is set up is that it uses P134 to do it. P134 itself uses JS2P10; the significant thing is that P10 must preserve an extra link (that from P52 to P134) in SHUL*, because this is part of the subprogram rather than the procedures outside it. Also the link parameter must be negative since EXIT 2 is required: so its value is actually -2.

Another interesting point is that, once the subprogram is re-entered, it may return to the same setting-up point (in P52 this is:- 3; SET -2; ZERO; ZERO; JSP134;) to suspend itself if a paper tape reader is not available. The same link parameter is used because the P52-P134 link must still be preserved in SHUL*, and it does not matter that P10 returns control to P11 using EXIT 2.

Note that when the subprogram suspends itself because there is no paper tape reader available, it does not use the convention of setting D0 of SHUL =1, and SHUP . D0+type number, because this does not give the operator any opportunity of using TINT A to terminate the input request (hence the use of P134). If TINT A occurs, the subprogram clears V2P52 and enters P121 (CRNP FO).

Until a paper tape reader is actually allocated and its number stored in V2P529 this V-store is used to hold an indication of whether the PTR is to be pre-allocated (status P) or not. If the unit is not pre-allocated it retains status U but the program letter is inserted in its descriptor to prevent it from being allocated for any other use.

It will be noted that, in these four examples, some other subprogram, besides 0, is involved. However the other subprogram is never set up until all the necessary checks have been performed - the failure of a check leads to EXIT 1 and the repetition of the TINT; query. After the new subprogram is set up (using an altered value of Q4) P10 transfers control back to P4 via EXIT 2; this resets Q4 and then returns to P11, leaving subprogram 0 closed down.

5. OUTS (P3, etc) Program Failure (P2)

When P3 is entered from the long path following an OUT, its first action is to clear V20P0, the OUT marker. Then it adds 3 syllables to the programs return address which is stored in V16P1, and checks that there is an "OUT number" in the right range stored at the top of the Nest, in V0P1. A copy of this is transferred to the SJNS, and this is used to perform a branch to one of a set of routines dealing with individual OUTS.

Unlike the branch following a TINT, this one is not performed like a "jump to subroutine", and the only item in the SJNS when one of these OUT routines has been entered is the link to return control from P3 to the long path. This link will be used by EXIT 1 at the end of an OUT routine, or at its first suspension,

At this stage the OUT number, in V0P1, has not been erased from the program's Nesting store, whose contents are in the V-stores of P1. The individual OUT routines are permitted to refer to the program's Nest in these locations, but only until the first suspension occurs, after which P29 must, of course, be used to refer to these items. Each OUT routine must make its own arrangements to remove the OUT number (and any other erasable parameters), and to insert "results", in the program's Nest. The convention is followed that should a program failure occur in one of these OUT routines, P2 (the Program Failure routine) will be entered after the OUT number has been erased from the programs Nest, but will find other parameters as they were on entry to Director.

Any OUT routine may want to initiate the main subprogram of the interrupted priority. Anticipating this, the long path will already have stored this priority number in M5, and using P30 will have set the correct parameters for this subprogram in Q4, before entering P3. Also, before branching to the appropriate OUT routine, P3 will set V19P0 (the EDT marker) non-zero, so that after P3 has returned control to the long path, P5, and hence P11, will subsequently be entered, giving the opportunity of re-entry to a suspended subprogram. By causing any OUT activity which involves a hold-up to be carried out by the appropriate subprogram, the Director ensures that the activity has the correct relative priority.

The quartets V45P104M5 and V49P104M5 can be used by these subprograms to hold parameters,

The OUT routines are:

OUT 0 - P15	OUT 6 - P21
OUT 1 - P16	OUT 7 - P21
OUT 2 - P17	OUT 8 - P23
OUT 3 - P18	OUT 9 - P24
OUT 4 - P19	OUT 10 - P61
OUT 5 - P20	

Before describing a few of them in detail, it is worth having a closer look at P2, the "Program Failure" subroutine.

5.1 P2 (Program Failure)

P2 is entered (using JSP2) from the long path, if a LIV, NOUV, Time Expiry, RESET or "Spurious Interrupt" failure occurs in a program; and also (using JP2) if a check fails in one of the OUT routines, or following a CRNP failure (P121) after OUT 1 - in these cases, there is already a link in SJNS which P2 can use to return control to the long path or to P11.

On entry to P2, M5 should contain the failed programs priority number, and Q4 the parameters for its main subprogram. There must be a failure indicator in N1, which is typed out either just as it stands (D0-38 non-zero), or, if D0-38 = 0, as the two-digit octal number in D39-44, followed by L, N, T according to which, if any, of D45, 46, 47 are non-zero (this is the same as the failure indicator required by /RW8/ in the non-Time Sharing Director - N7.1). If program termination is unconditional, V0P2 must contain a non-zero "Ending number"; otherwise the familiar "REACT;" -query will be typed later, and termination only occurs if the answer to this query is "A".

P2 has a second entry point at reference 2, which occurs after the failure indicator has been typed. This entry is used by TINT A (P33 - see 4.2.1), which sets the ending number "4" in V0P2 before obeying J2P2. At this point P12 is entered to type out the, contents of the programs Nest and SJNS. P12 uses P29 for all references to the Nest and SJNS, except when V6P12 is non-zero: this only occurs following a NOUV interrupt, when it is certain that the contents of the Nest and SJNS are still in the V-stores of P1.

After being typed, the program's Nest and SJNS are cleared. If V0P2 is non-zero, this ending number is set in N1, and control now passes to P15 (see 5.2.1) using J1P15.

If V0P2 is zero, as is the case for most program failures, the "REACT;" query is typed using the priority's main subprogram. Because this query has to be typed in a different form for each program, it is stored in its four different forms in the V-stores of P2, thus:

V2 - 4 - Program P
V5 - 7 - Program Q
V8 - 10 - Program R
V11 - 13 - Program S

This avoids falling into the trap mentioned in 4.1.3.

The query is typed using P8 with a zero link parameter. If the first character of the reply (in V4, 7, 10 or 13 of P2) is "A", the program must be terminated - the ending number 7 is set in N1, and control passed to P15, using J1P15. Otherwise, the value 4 is stored in the program's SJNS (to act as the re-entry address), and the programs time limit is replaced by its run time to date plus 4 seconds. P2 ends by obeying EXIT 1, thus closing down the main subprogram.

5.2 OUT Routines

All these routines act as extensions of P3, and, in the case of P15, of P33 (TINT A), and P2 (Program Failure). All are expected to end EXIT 1, and therefore to use a positive link parameter at any subprogram suspension.

P23 has features intended for use by all OUTs, present and future, which perform peripheral transfers on a programs behalf, and is discussed separately in section 5.3.

Before describing individual routines, mention should be made of the useful subroutine P62. This is used before any suspension occurs. It erases the OUT number from the programs Nest, checks that the latter contains at least one more item, and leaves two copies of that next item in N1 and N2, without erasing it.

5.2.1 P15 (OUT 0) and P17 (OUT 2)

P15 has 3 entry points. The initial actions of P15 are described below, with the entry points indicated.

- main entry (JP15 - used by OUT 0): set ending number of zero in N1;
- second entry point (J1P15 - used by P2 and P33): given the ending number in the Nest, sets 5 (the termination marker, to be stored in V29P104M5), in N1;
- third entry point (J2P15 - used by P17 (OUT 2)): at this stage the ending number is in N2 and the termination marker is in Nl. The latter has the value 5 unless termination is by OUT 2, in which case it is 8 (A-program) or 9 (B-program).

From this point the important actions of P15 are:-

- to add D0 to the termination marker, store it in V29P104M5, and set D46 of V25P104M5 non-zero.
- to dispose of peripheral units. Any unit allocated or pre-allocated to a program has its current transfer cleared. If it is a magnetic tape, a "block-counting" rewind is performed. Units of other types revert to status U, unless OUT 2 is being performed for an A-program, in which case these units revert to status P.

  Note that it is possible to find a unit in status A even after the program has obeyed OUT 6 to deallocate it, because the unit must stay in that status until it is no longer busy (the program being prevented from continuing by reason of a suspension of its main subprogram). If, while the unit is still busy, TINT A is executed, the subprogram is forcibly closed down, the unit remains in status A, and, if it is still busy, will have its transfer cleared. Thus there is a slight danger that if a busy unit is de-allocated by OUT 6, the transfer may still be cut short if TINT A occurs before it has finished. This could not

  arise with the non-Time-Sharing Director because no TINT could be performed while the Director was waiting for the transfer to end.

- if the second (OUT 8) subprogram of this priority is closed down, i.e. if E1M4 is zero, P15 sets it up again so that its SHUP (E11M4) and SHUL* (E31M4) are zero, and its SHUL (E1M4) contains a (positive) link causing it to be entered at reference 99 of P23 (see 5.3).

  P15 now suspends itself (using JS4P10) until V29P104M5 is positive. It will be recalled that D0 of this word was made non-zero at the third entry point to P15. This is interpreted by the second subprogram as a signal that the program is to be terminated and accordingly it winds up all OUT 8 - type processes. When it has done this and the programs store area is free of lock-outs, it clears D0 of V29P104M5 and closes itself down.

  P15 can now resume and it types out on the log the details of the last program. If OUT 2 is in progress, P15 next modifies the program's NOL if required, using a value left by P17 in V45P1O4M5, and ends by obeying J44P120 - this causes P120, the program input routine, to be joined at the point where the details of the new program are typed on the log, and, inter alia, V29P104M5 and D46 of V25P104M5 are cleared before EXIT 1 is obeyed, closing down the main subprogram.

  If not initiated by OUT 2, P15 obeys J10P121 - this transfers control to the last stage of the CRNP failure routine, in which BA and NOL (V9P0M5) are cleared, the "program present" bit is removed from V0P104, the area the program occupied is checked to ensure no lockouts remain, and that priority's PHU store is cleared. EXIT 1 is obeyed to close down the main subprogram. V29P104M5 is left with the value 5 in it.

P17 (OUT 2) first carries out various checks on the new program identifier, store size, etc. before setting an ending number of 2 and a termination marker value of 8 (A-program) or 9, and obeying J2P15. Any failure will lead to P2, an ending number of 6 first having been placed in V0P2.

One interesting feature is the preservation of the contents of D24-47 of E0 in V33P104M5. This is a buffer used by TINT B, if executed while a program is in process of termination, and its contents will be restored to E0 by P120 when it is eventually entered at reference 44.

The value of NOL for the new program is left by P17 in V45P104M5, to be used by P15.

5.2.2 P16 (OUT 1)

After various checks, the value 2 (A-program) or 3 is set in V29P104M5, and D46 of V25P104M5 made non-zero. The main subprogram is suspended if the store area of the program is locked out, if the second (OUT 8) subprogram is not closed down, or if V0P120 (A program)/V0P52 (B program) is positive, and will not be resumed until all these conditions are absent On resumption, the priority number will be stored in V0P120 or V0P52, to indicate that A- or B-program input is going on.

Before this suspension a message "OUT 1" is typed: the operator will be aware that the suspension has not been lifted if the new program identifier is not typed subsequently.

Finally P16 obeys J22P120 to transfer control to the program input routines. Parameters for P120 will have been left in V1P104M5 (the amount of store used by the previous section, not to be exceeded by the new one) and in V45P104M5 (the identifier of the required new program section).

Any failures after entry to P120 will of course be dealt with by P121 (CRNP failure), which recognises that the program input was initiated by OUT 1 (since D46 of V29P104M5 is non-zero) and accordingly transfers control to P2 - having first set the ending number 5 in V0P2 to guarantee termination.

5.2.3 P18 (OUT 3) and P24 (OUT9)

These two very simple routines are examples of the way in which OUT routines may, until their first suspension, rely on the program's Nest staying in the V-stores of Pl. The appropriate times (Run or Real) are transferred to V0P1 (N1 of the program's Nest). Thus the OUT number is erased and the result inserted with just one instruction.

5.2.4 P19 (OUT 4) and P61 (OUT 10)

These routines both, after performing all the necessary checks, use P58, whose action has been described in 4.1.6. If the required identifier is one-word, an artificial second word is introduced which has D0-2 0, D2-47 = all l's.

A marker is set in V45P104M5 to distinguish OUT 4 (marker non zero) from OUT 10 (marker zero).

5.2.5 P20 (OUT 5) and P21 (OUT 6). P26

The behaviour of the OUT 5 routine depends largely on whether the program obeying OUT 5 is A or B. An A program can only be allocated units of status P, and program failure results if there are none available. A B program will be allocated a P unit if there is one (a unit may be pre-allocated to a B program either as the result of TINT T, which can pre-allocate the input paper

tape reader, or following the de-allocation of a unit previously allocated by OUT 5*); if there is no P unit of the required type, a U unit must be allocated and if there is not one of these the "AWAITS TYPE tt" message is typed and the subprogram suspended (with SHUL negative, and SHUP = type number + D0). This suspension is cleared either by a unit of the right type becoming available, or by TINT A or I0.

OUT 5* (when the type number supplied by the program in N2 has D44 added) is dealt with as follows:- When a B-program asks for a unit of a certain type in this way, P20 inserts a marker bit in V21P104M5. The bits of this word are assigned to type numbers in such a way that type (16a + b) is represented by D(48 - 8a - b). When a B-program uses OUT 6 to de-allocate a unit whose type bit in V21P104M5 is non-zero, P21 clears that bit and the unit is given status P, rather than U as would normally happen. If a B-program tries to use OUT 5 or 5* to allocate a unit, specifying a type whose bit is non-zero, Program Failure 2 occurs even if a unit of the required type is available. The system. thus has exactly the same effect as in the non-Time-Sharing Director.

This only applies to B-programs. Because any unit (not magnetic tape) which is de-allocated by an A-program is given status P, and only one unit of each type is pre-allocated, OUT 5 and OUT 5* are identical for A-programs. For an A-program V21P104M5 instead contains a record of all the units pre-allocated to the program: D47 = unit 0, D46 = unit 1, D37 = unit (octal) 12, etc.

This correspondence of bits with unit numbers, which is the same as that used in CPDAR (N2.1.4), also applies to the copy of the programs CPDAR kept in V17P104M5, which is updated by P20 and P21 as units are allocated and de-allocated.

When de-allocating a magnetic tape unit, P21 sets a brick indicator and uses JS10P100 (see 4.1.1) to alter the unit's status to W. For any other type of unit, P21 has to wait until the unit is not busy before changing its status to P or U. Therefore if the unit is busy the subprogram is suspended, with SHUL positive and the unit number in D0-15 of SHUP, The end of such a suspension will not usually be signalled by an EDT and so may have to wait for an EDT from some other source. As has already been pointed out (5.2.1) if TINT A is used to terminate the program while this suspension is still present, the unit, if still busy, may have its transfer forcibly terminated.

The "descriptor" of unit n is stored in Vn of P26. Each descriptor is laid out as follows:

```
D0  - 11 = (octal) 0207
D12 - 23 = Type number (octal) in character form
D24 - 29 = Status character
D30 - 41 = Unit number (octal) in character form
D42 - 47 = program letter if any
```


P26 is used for finding a unit or units having a particular type, status and program letter. It has 3 entries and 2 exits. The first 2 entries (JS1P26, JS2P26) require in N1 a parameter comprising:-

```
D33 - 38 - required type number
D39 - 44 - required status character
D45 - 47 = 4 if program letter is to be blank
         = 0, 1, 2, 3 if P, Q, R, S.
```

The list is searched backwards, i.e. through decreasing unit numbers. The first entry (JS1P26) is used if the search is to start at the highest unit number, the second entry (JS2P26) if the search is to resume after the last unit found (i.e. with Q6 unaltered since the last EXIT 2). EXIT 1 is obeyed if no unit was found: the parameter in N1 will have been erased. EXIT 2 is obeyed after a successful search, when C6 and M6 will both contain the number of the required unit, with I6 = -1. After EXIT 2, N1 will contain a modified version of the original parameter, and if desired the search may immediately be resumed, without re-specifying the original parameter, by obeying JS4P26.
(Compare P26 with /RPS1/ and /RPS2/ of the non-Time-Sharing Director - N5.3).

5.3 P23 and OUT 8

All transfers executed by the Director on behalf of a program (OUT 8 is the obvious example) have certain features in common. A system has therefore been evolved into which additional facilities of this type can be fitted without too much difficulty, and the framework of this system forms part of P23, the OUT 8 routine.

Besides using the main (even) subprogram, the system makes use of the second (odd) subprogram of the priority instigating the transfer - i.e. for priority p, subprograms 2p + 2 and 2p + 3. The second subprogram is used because these transfers, once started, do not require the program to be held up - the transfer area being protected by ordinary lock-outs - but do require that when they end, a procedure be used which not only checks the transfer but also signals that a further transfer may now take place, For this procedure the main subprogram is not suitable, since it would hold up the program unnecessarily, but the second subprogram is ideal.

All Director transfers of the OUT 8 type pass through 4 stages The first stage involves erasing the OUT number from the program's Nest and checking that the parameters provided by the program in its Nest are valid - e.g. that the addresses are reasonable. It is assumed that OUTS which perform this sort of transfer will supply all parameters in N2, and that these will implicitly or explicitly specify the relative addresses of the transfer area.

This stage will end after these parameters have been copied into V45P104M5 (and possibly V49P104M5) without erasing them from the programs Nest, and the initial and final absolute addresses of the transfer area into I7 and M7. P133 (q.v.) is a useful subroutine for this purpose - it also checks that these addresses are valid. The first stage ends by obeying JSP139.

The subroutine P139 causes the main subprogram to be suspended until the area involved in the transfer is not locked out, and the second subprogram is closed down (E1M4 = 0). When both these conditions are satisfied, the second stage can begin. In this stage the Director, still acting through the main subprogram, is able to check, and if necessary alter, the transfer area itself: and, because the second subprogram has been closed down, it may check any failure indicators (e.g. parity) left by the previous transfer of this type, which must now be over. It is a feature of the Time-Sharing Director that each priority may only carry out one transfer of this type at a time, because each transfer requires the use of the second subprogram.

Causes of program failure (JP2) may be found in the second as well as in the first stage. At the end of the second stage the main subprogram is ready to initiate the second subprogram, which is going to actually perform the transfer. To do this the main subprogram must obey J97P23 with certain parameters in the nesting store. They are:-

N1 - SHUP* of second subprogram. This is to contain, in D16-31 and D32-47, the initial and final <u>relative</u> addresses of the transfer. D0-15 may contain any other parameter.

N2 - SHUL of second subprogram, defining its point of entry.

N3 - a marker which is zero if the transfer is going to be queued, nonzero otherwise.

On entry at reference 97, P23 transfers N1 and N2 to SHUP* and SHUL of the second subprogram (E21M4 and. E1M4), and clears SHUL* (E31M4) and SHUP (E11M4) - the latter guarantees that at the very next opportunity, i.e. as soon as the main subprogram returns control to P11, the second subprogram will be entered. The main subprogram now obeys JS3P29 to erase the parameter word (originally in N2) from the programs Nest, and returns control to P11 - it does this by obeying either EXIT 1 (thus closing down the main subprogram altogether) if the transfer is not to be queued, in which case D44 of V25P104M5 is first made non-zero, or by obeying JS4P10, if it is queued.

Stage three of the process takes place in the second subprogram. This will extract the parameters from SHUP*, make the addresses absolute, and initiate the transfer, either by entering it in a queue or by using an ordinary transfer instruction. If the latter, D44 of V25P104M5 is cleared as soon as the transfer is started, since this will cause lock-outs to be set up on the transfer area, making it safe to allow the program to resume; the second subprogram is suspended until the end of the transfer, and the first subprogram is already closed down. However, if the transfer is queued there is no guarantee that it will start immediately, so it is not yet safe to return control to the program. It will not in general be possible for the second subprogram to hold up the program only until the lock-out is set up, because it will itself probably be suspended until the very end of the transfer. This is why, in this case, the main subprogram is not closed down; it remains active (thereby holding up the program) and

only closes itself down when either the area becomes locked out, or the transfer parameters (taken from E21M4, which must not be altered by the second subprogram) are found to be absent from all the peripheral queues. The main subprogram repeatedly tests these conditions, suspending itself with JS4P10 after each test cycle, until at last it obeys EXIT 1 when it is safe to return to the program. It is possible that a TINT A or TINT I0 may close down the main subprogram before this time: in these special circumstances this does not matter, and the second subprogram is not affected.

The second subprogram, continuing the third stage of the process, must remain active until the transfer is complete and checked (after repetition if need be). If it is necessary for this subprogram to hold up the program it can reset D44 of V25P104M5. The parameters in its SHUP* must not be disturbed if the transfer is queued. Even if no check on the working of the transfer is needed at its end, the ending of the third stage must be delayed until the transfer is complete; so that, if the program is terminated before the transfer ended, it will still be possible to indicate definitely when all lock-outs on the program's store area should have been removed.

The fourth and final stage of the process is entered by obeying J99P23. At this point (at which P15 will restart the second subprogram if it finds the latter already closed down see 5.2.1) V29P104M5 will be examined: if it is positive, the second subprogram can be closed down, but if it is negative, indicating that the program is being terminated, a winding-up procedure for all OUT 8 type activities must first be performed, culminating in D0 of V29P104M5 being made zero again.

Before the second subprogram is closed down, D44 of V25P104M5 is cleared. Also, since the main subprogram may be suspended in P139 waiting for the closing-down of the second subprogram, the subprogram parameters in Q4 are adjusted as if they referred to the last subprogram but one; in this way, when EXIT 1 is obeyed to close down the subprogram and return to P11, the latter will now try to re-enter the main subprogram. In other words the subprogram counter is effectively stepped back one place instead of forward when the second subprogram is closed down.

The common features of all these OUT - instigated peripheral transfers are thus:

- a first stage, leading to P139
- a second stage leading to 97P23

(these two stages carried out by the main subprogram)

- a third stage, carried out by the second subprogram, which performs the transfer, and leads to
- the final stage, entered at 99P23, leading to the closing down of the second subprogram (and reverting to the main subprogram)

To fully appreciate the way these common features are exploited, it is worth examining and comparing the coding for the different types of OUT 8 process in P23, and for the OUT 11 - OUT 12 (drum transfer) routines P151 and P152.

6. Subroutine Index

The table below gives against each subroutine a brief title, the number of the section (if any) of the preceding text in which the subroutine is described most fully, and, where applicable, an indication of which Q-stores it alters. A single asterisk against a subroutine indicates that it requires M5 preset with a priority number; a double asterisk, that Q4 is to be preset with subprogram parameters.

```
  P0       Long and Short. Paths (3.1)
  P1       Dump programs Nest, etc. (3.3)
**P2       Program Failure (5.1)
**P3       OUT (5)
  P4       FLEX (4.2)
  P5       EDT  (4.1)
  P6       Enter Message words in Flexowriter queue (4.1.3)
           Alters Q7, C5
  P7       Enter Parameters in Flexowriter queue (4.1.3)
           Alters Q7, C5
**P8       Flexowriter query (4.1.3)
 *P9       Convert cell count, used by P12
**P10      Suspend subprogram (4.1.6)
  P11      Re-enter subprograms in order (4.1.5)
 *P12      Type program's Nest, etc. (5.1)
           Alters Q6, Q7
 *P13,P14  Set and clear "software HU" bit in V0P104
           Alters C5
**P15      OUT 0  (5.2.1)
**P16      OUT 1  (5.2,2)
**P17      OUT 2  (5.2.1)
**P18      OUT 3  (5.2.3)
**P19      OUT 4  (5.2.4)
**P20      OUT 5  (5.2.5)
**P21      OUT 6,7(5.2.5)
**P23      OUT 8  (5.3)
**P24      OUT 9  (5.2.3)
 *P25      Check program A or B.
           Alters C6
  P26      Find unit with given type, status and program letter (5.2.5)
           Alters Q6
 *P27      Alter unit status on de-allocation
           Alters Q7
 *P28      Look for P unit
           Alters Q6
 *P29      Manipulate program's Nest, etc. (3.3)
           Alters Q6
 *P30      Set up main subprogram parameters in Q4 (4.2.1)
  P31      Fetch character to 07 (4.2)
           Alters Q6 Q7
  P32      Fetch number (4.2)
           Alters Q6, Q7
**P33      TINT A (4.2.1)
**P34      TINT B (4.2)
```



```
**P37    TINT E (2.5)
**p39    TINT G (4.2)
**P40    TINT H (4.2)
**P41    TINT I (4.2.2)
**P42    TINT J (4.2)
**P44    TINT L (4.2)
**P45    TINT M (4,2,3)
**P48    TINT P (4.2) (also used in Phase 0)
**P50    TINT R (4.2)
**P51    TINT S (4.2)
**P52    TINT T (4.2.4)
**P53    TINT U (4.2)
**P54    TINT V (4.2)
 *P55    Obtain program letter, given priority number
  P56    Obtain priority number given program letter in C7 (402)
         Alters Q5
  P57    Convert time for typing
**P58    Find tape with given identifier (4.1.6)
  P59    Update Flexowriter queue (4.1.3)
         Alters Q7, C5
  P60    Deal with RPM (303)
         Alters no Q stores
**P61    OUT 10 (5.2.4)
 *P62    Initial housekeeping for OUTS (5.2)
         Alters Q6
  P100   Controller of W mag tape bricks (4.1.1)
  P101   W mag tape bricks (4.1.1)
**P102   Hold up subprogram before transfer
  P103   Check before transfer (2.4)
         Alters Q6, Q7
  P104   Priority parameters (204)
  P105   Consolidate store and swop priorities (4.1.2)
  P106   Set up Q7 with program parameters
         Alters Q7
  P107   Test lock-out on programs area
         Alters Q7
  P108   Test whether given transfer area lies within
         area specified in Q7
  P109   Modify transfer parameters after store move
  P110   Alter priority number if necessary after swop
  P111   Set up priority swop
  P112   Store new BA/NOL

  (P106 - 112 are primarily used in P105)

 *P113,P114 Set or clear given bit in V25P104M5 (2.4)
  P115   Set priority related parameters in C6 and M6
         Alters C6, M6
  P116   Form table of BA/NOL words in order of decreasing BA
         Alters Q6, Q7
  P117   Form parameters of the "Hole in the Middle"
         Alters Q6, Q7
 *P118   Set up parameters before program input (4.2.4)
         Alters C5,M6
```



```
  P119   Check program name
  P120   Look for A input priority (4.1.4)
 (later entries: ** general program input)
**P121   CRNP failures (2.5)
  P122   Check tape identifier
         Alters C7
 *P123   Dispose of program input PTR
         Alters C7
 *P124   Set AV0P52 or AV0P120 in M6 (2.5)
         Alters M6
 *P125   Check validity and ownership of peripheral unit
 *P126   Set addresses of E0 and E7 in 17, M7
         Alters 17, M7
  P127   Check for parameter word in Flexowriter queue
 *P128   Makes absolute addresses in Q7
         Alters I7, M7
  P129   Checks given parameters to see if still in Flexowriter queue
         Alters Q6
  P137   Leaves 3 times M6 in N1
  P132   Check length of identifier given first word
 *P133   Checks transfer parameters in first stage of OUT 8 process (5.3)
         Alters Q7
**P134   Hold-up program input so it can be terminated by TINT A (4.2.4)
  P135   Dispose of program tape
 *P136   Store new BA/NOL, check lock-outs, clear PHU
         Alters Q7
  P137   Compare N1 and N2, ignoring octal 36 in N1
         Alters C6
 *P138   Alter bit in V17P104M5 (copy of CPDAR)
**P139   End of stage 1 of OUT 8 process (503)



  P200   "Phase 0" (N 3.1)
  P201   Type message in Phase 0
  P202   Type query in Phase 0
         Alters Q5, C6
  P204   Read decimal number in Phase 0
  P206   Type unit list in Phase 0
         Alters Q5, Q6
  P208   Read OUT 8 reel number in Phase 0
         Alters Q5, Q6
 (Subroutines P200 onwards are overwritten as soon as Phase 0 ends)
```